

# Duplicate Detection with Efficient Language Models for Automatic Bibliographic Heterogeneous Data Integration

Nicolas Turenne\*

Univ. Paris-Est, LISIS, INRA  
F-77450 Champs-sur-Marne, France

## Abstract

We present a new method to detect duplicates used to merge different bibliographic record corpora with the help of lexical and social information. As we show, a trivial key is not available to delete useless documents. Merging heterogeneous document databases to get a maximum of information can be of interest. In our case we try to build a document corpus about the TOR molecule so as to extract relationships with other gene components from PubMed and WebOfScience document databases. Our approach makes key fingerprints based on n-grams. We made two documents gold standards using this corpus to make an evaluation. Comparison with other well-known methods in deduplication gives best scores of recall (95%) and precision (100%).

Keywords: fingerprint algorithms; shunks algorithms; information retrieval algorithm; deduplication; collocations; n-grams; natural language processing; database cleaning

## 1 Introduction

Since early 90ies with internet and low-cost workstation development, lots of large public or commercial databases emerged. Sometimes fusion of data from different databases seems to be a good strategy to build a convenient document repository about a specific topic. In this way cleaning and normalization is necessary. If deduplication between documents from a same

---

\*nturenne.inra@yahoo.fr

data repository is trivial (through title or identifier matching), indexing over databases is not the same and just comparing whole title is not sufficient (see below for examples). In this paper we are interested in deduplication (also called dedupes) between two specific databases WebOfScience (commercial) and PubMed (public). There are two tasks for such data processing. First, it can be understood as the elimination of redundant data in computer storage (see [35], [17], [4], [15]). Second, it is related to record linkage, in databases: that is, finding entries that refer to the same entity in two or more files (see [26],[13]). Most duplicate detection methods can be classified as:

1. domain-dependent: requiring some knowledge of the data source. For instance, data fields.
2. domain-independent: not requiring any knowledge of the data source. Generally this uses text strings within the records to search for matches.

Our goal is to merge records from bibliographic documents not knowing the structure. Some approaches try to detect near-duplicates aiming at identifying plagiarism (see [18],[9],[14],[11]). Others focus, as our approach, on detecting exact duplicates, so as to merge or delete entries when merging items between heterogeneous databases (see [29],[22],[24],[34],[27],[25]).

If a primary key exists, the duplicate records could be easily identified. Such a key exists for a document database and is called a DOI (Document Object Identifier). It looks like

*10.1016/j.bcp.2010.01.019.*

Its use leads to visualization of the document's electronic version on the Internet. But the use of such a key is recent and building useful corpora including documents published before 2005 makes this approach irrelevant. Comparing two documents can be, in the case of short texts, a string alignment. Some works have led to interesting processing techniques such as the Levenshtein (or edit) distance [21], based on a cost computation for the deletion and insertion of characters to transform one string into another; or fast algorithms in the tradition of [20] and [5], in the area of DNA sequences alignment these are well described by [31] and [1]. All current techniques for solving the duplicate discovery problem in a document collection are based on document fingerprinting, in which a compact representation of a selected subset of contiguous text chunks occurring in each document-its fingerprint-is stored. Pairs of documents are considered as possible duplicates if enough of the chunks in their respective fingerprints are matched. A chunk is defined [7], as a contiguous subsequence; that is, a chunk represents a contiguous

set of words or characters within the document.

Section 2 presents the data and our application in molecular biology. In Section 3, we present the state of the art of deduplication and some definitions. In Section 4 we introduce our language model methodology of fingerprinting. Finally in Section 5 we show the results of evaluation and comparison with six other approaches: three fingerprinting approaches (simple fields-key method, multi-fields-key method, anchor method), two scoring approaches (Salton information retrieval method, collocation method), and the random approach (Monte Carlo method).

## 2 The biological issue and document collections

In this section, we present how and for what the document collections were built.

### 2.1 The biological issue

The background issue concerns understanding how the TOR molecule is involved in terms of upper and down regulation of other genes in the framework of a species (such as the plant *Arabidopsis thaliana*). TOR gets lots of interest in the research community because of its role in growth and metabolism. Comparison with other species is a knowledge tool often used by biologists. TOR has similar genes (called orthologs) to other species (mouse, human, yeast, worm and fly). A corpus covering all possible links between TOR and other molecules (TOR-interactomes) could be of great interest, and secondly, the interaction between components of two TOR-interactomes could also brings some pieces of knowledge. It should be interesting to detect equivalent genes using existing databases and the BLAST approach (DNA sequence comparison) but databases are not always well annotated. Usually the NCBI pubmed is viewed by biologists to search documents and even biological data. Many others sources of documents are widespread. A famous one is Web of Science. It claims encapsulation of pubmed but updates are not running well. Our idea for getting a larger corpus of documents relies on the fusion of data collected from different databases to permit the maximum extraction of relevant relationships between gene associations.

### 2.2 Document collections (corpora)

We used for convenience two well known databases to build our corpora. The first one is PubMed, indexing more than 19 million documents with

an update of 3,000 documents/day. And the Web of Science indexing more than 90 millions documents with an update of 5000 documents per day. Since a couple of years ago, Web of Science has been integrating PubMed.

A manually defined query based on TOR variants names has been applied to both databases (PubMed, hereafter PM, and the Web of Science, hereafter WOS).

```
query: (raptor[Title/Abstract] OR kog1[Title/Abstract] OR lst8[Title/Abstract]
OR "target
of rapamycin"[Title/Abstract] OR TOR[Title/Abstract] OR TORC1[Title/Abstract]
OR
TORC2[Title/Abstract] OR mTORC1[Title/Abstract] OR Dd-TOR[Title/Abstract]
OR
mTORC2[Title/Abstract] OR TOR1[Title/Abstract] OR TOR2[Title/Abstract]
OR
mTOR[Title/Abstract] OR dTOR[Title/Abstract] OR CeTOR[Title/Abstract]
OR
AtTOR[Title/Abstract] OR Tor1p[Title/Abstract] or Tor2p[Title/Abstract])
not "tor vergata"
```

The result of the query is then cleaned by excluding documents from the following topics.

Refined by: [excluding] Subject Areas=( Meteorology & Atmospheric Sciences OR Telecommunications OR Education & Educational Research OR Astronomy & Astrophysics OR Instruments & Instrumentation OR Nursing OR Computer Science OR Social Issues OR Communication OR Radiology, Nuclear Medicine & Medical Imaging OR Water Resources OR Optics OR Nuclear Science & Technology OR History OR Information Science & Library Science OR Geochemistry & Geophysics OR Energy & Fuels OR Polymer Science OR Mathematics OR Business & Economics OR Government & Law OR Geography OR Sport Sciences OR Anthropology OR Sociology OR Engineering OR Mechanics OR Materials Science OR Automation & Control Systems OR Physics OR Family Studies OR Psychology OR Geology OR Imaging Science & Photographics Technology )

An Overview of corpora size and number of some important fields are shown on Figure 1.

### 2.3 Duplication issue

As mentionned in the Introduction, when a unique identifier is attached to a document, its usage is immediate to avoid any duplicated insertion during

	#docs with filtering	#docs	#words	year range	#Field Author	#Field Title	#Field Source
<b>PM</b>	7,709	7,709	3,870,000	1951-2010	7,709	7,689	7,709
<b>WOS</b>	12,658	16,178	2,780,000	1951-2010	12,658	12,658	12,658

Figure 1: Statistics about corpora content.

a database fusion operation.

Such an identifier does not systematically exist for all documents and moreover we observe that some differences for the same document can occur in the title.

For instance the sentence below belongs to the same publication but is indexed differently in PM and WOS databases:

1. PM sentence: *Here we show that CRP is required for the biosynthesis of cholera autoinducer 1 (CAI-1) (PMID-17768239)*
2. 'WOS' sentence: *Here we show that CRP is required for the biosynthesis of cholera autoinducer 1 (CAI-1) (UT WOS:000250013400014)*

They contain a different writing of the gene name cai-1.

This is another example:

1. WOS sentence : *Structural Analysis and Functional Implications of the Negative mTORC1 Regulator REDD1 (UT WOS:000275711400021)*
2. PM sentence : *Structural analysis and functional implications of the negative mTORC1 regulator REDD1. (PMID- 20166753)*

In this case the gene name mTorc1 is written with l in PM instead of 1 in the WOS

A last example, insertion or permutation of characters can occur as in :

1. WOS sentence : *Bis(morpholino-1,3,5-triazine) Derivatives: Potent Adenosine 5'-Triphosphate Competitive Phosphatidylinositol-3-kinase/Mammalian Target of Rapamycin Inhibitors: Discovery of Compound 26 (PKI-587), a Highly Efficacious Dual Inhibitor (UT WOS:000275805900058)*
2. PM sentence : *Bis(morpholino-1,3,5-triazine) derivatives: potent adenosine 5'-triphosphate*

*competitive phosphatidylinositol-3-kinase/mammalian target of rapamycin inhibitors: discovery of compound 26 (PKI-587), a highly efficacious dual inhibitor.* (PMID- 20188552)

where 5'-Triphosphate is cut after 5 in the WOS document. When reconstructing the sentence, the cut is replaced by a blank and makes matching impossible.

The duplication issue is a two-class problem where classes are 'duplicate' tag and 'non-duplicate' tag. A test-database is processed by an algorithm. Such algorithm picks up each document from the test-database to transform it and compare to each document, transformed in the same way, of a target-database. For our purpose, the test-database is the PM database and the target database is the WOS database.

### 3 state of the art of deduplication and some definitions

In this section, we present some definitions about linguistic parsing and the best algorithms for deduplication.

#### 3.1 Definitions about formal content of a document

The approach we adopt to develop, usually concerning the scientific community, relies on a definition of a language model to compare document contents.

Firstly we need to define what is an elementary alphabet.

**Definition 3.1** Alphabet

Let  $A$  be the set of characters  $[a,b,c,\dots,z;0,1,\dots,9]$  to make a word; some special characters can belong to a word such as  $[-]$ .

Content description requires some delimiters:

**Definition 3.2** Delimiter

Let  $D$  be the set of special characters  $[space =,;: () * +.,\%[]]$ ; the *space* character is the main delimiter.

Then we can define general string.

**Definition 3.3** String

A string  $S$ , of length  $N$ , is a set of words  $\{W_i\}_{i=1,N}$  separated by one or several delimiter(s).

At the word level, interesting patterns in a string could be elementary sequences we call a n-gram:

**Definition 3.4** n-gram

A n-gram is a set of  $n$  consecutive character(s)  $\{C_j\}_{j=1,n}$  in a string where  $C_j$  belongs to  $A$ .

In documents, useful components are described by associations. These associations, we call n-collocations, defines semantic links:

**Definition 3.5** n-collocation (or chunk)

An n-collocation is a string of  $n$  consecutive words  $\{W_i\}_{i=1,n}$ .

Finally at the document scale, a document is divided into several fields having different kinds of descriptors such as dates, authors, titles, sources of publication for instance:

**Definition 3.6** Fields

A bibliographic document is composed of a set of fields  $F_k$ ; each  $F_k$  begins with an identifier and is followed by a string.

Some algorithms requires extraction of a key:

**Definition 3.7** Key

A Key  $K$  attached to a given document is a language model, defined by the alphabet and/or strings and/or collocations and/or n-grams included in some fields of the document.  $K$  can be built for any document contained in a document database, and  $K$  is supposed to be unique.

**3.2 Main approaches**

Two families are well applied for deduplication: fingerprint approaches and similarity-based approaches (see [10], [28]).

Let us consider, to begin with, the following fingerprint approaches.

[16] proposed the sorted neighbourhood. This is done in three phases.

---

**Algorithm 1** Sorted Neighborhood (SN) algorithm

---

- 1: *Create a Key for each record*
  - 2: *Sort records on this key*
  - 3: *Merge/Purge records*
- 

Considering our purpose for comparing documents, it is convenient to adapt and derive some interesting variants of "Algorithm 1" using some fields. Some specific fields can serve directly or be combined as a key to make a fingerprint. A first variant, we will call it author-fingerprint (AF), make the key with the name of the first author. A second variant can be the selection of the title as a key, we call it title-fingerprint (TF). Based on the previous we can lightly process the title by deleting a delimiter such as space, we call this variant modified-title-fingerprint (MTF). Finally we can imagine, knowing what type of document is a scientific document, that meaningful combined fields should be first author, date and source of publication, so the key would be the concatenation of these fields; we call this approach author-review-date fingerprint (ARDF).

Let suppose the following example :

Let consider a real document published with the following fields (Authors, Title, Source of publication and Date):

- AU - Ayral-Kaloustian, S Gu, JX Lucas, J Cinque, M Gaydos, C Zask, A Chaudhary, I Wang, JY Di, L Young, M Ruppen, M Mansour, TS Gibbons, JJ Yu, K
- TI - Hybrid Inhibitors of Phosphatidylinositol 3-Kinase (PI3K) and the Mammalian Target of Rapamycin (mTOR): Design, Synthesis, and Superior Antitumor Activity of Novel Wortmannin-Rapamycin Conjugates
- SO - JOURNAL OF MEDICINAL CHEMISTRY
- DP - 2010

The trivial keys (K) with the previous methodologies should be, after transforming the strings to lower case:

1.  $K_{AF} = \text{ayral-kaloustian}$
2.  $K_{TF} = \text{hybrid inhibitors of phosphatidylinositol 3-pinase (pi3k) and the mammalian target of rapamycin (mtor): design, synthesis, and superior antitumor activity of novel wortmannin-rapamycin conjugates}$



3.  $K_{MTF}$  = hybridinhibitors of phosphatidylinositol 3-kinase (pi3k) and the mammalian target of rapamycin (mTOR): design, synthesis, and superior anti-tumor activity of novel wortmannin-rapa mycin conjugates
4.  $K_{ARDF}$  = aryal-kaloustian-journal of medicinal chemistry-2010

Sort data has time complexity  $O(N \log N)$  for a good algorithm,  $O(N^2)$  for a bad algorithm. To optimize at step 3 of "Algorithm 1", move a fixed size window through the sequential list of records. This limits comparisons to the records in the window. The multi-pass variant (SNM) involves using multiple passes of the initial SN.

[12] developed the k-way sorting method for no uniquely distinguishing data fields. The concept behind the k-way sort method follows (see "Algorithm 2").

---

**Algorithm 2** k-way algorithm

---

- 1: Let  $k$  be number of columns to be used for sorting
  - 2: Select the  $k$  most meaningful combination
  - 3: Assign a record identifier to each record
  - 4: Sort records
  - 5: Compare adjacent rows :
  - 6: **if** matched columns  $>$   $k/2$  columns **then**
  - 7:     the two records match
  - 8: **end if**
- 

(see "Algorithm 3") describes basic fingerprinting process is as follows (see [2], [30], [23]):

---

**Algorithm 3** General fingerprinting algorithm

---

- 1: documents in a collection are parsed into units (typically either characters or individual words)
  - 2: representative  $n$ -collocations are selected through the use of a heuristic
  - 3: the selected chunks are then hashed for efficient retrieval and/or compact storage
  - 4: the hash keys, and possibly also the chunks themselves, are then stored, often in a inverted index structure
  - 5: Compare co-derivatives documents using index of hash keys
-

The principal way in which algorithms differentiate themselves is in the choice of selection heuristic, the method of determination which chunks should be selected for storage in each document's fingerprint.

[19] envisioned an entity grouping approach for identifying author-title clusters in bibliographical records.

---

**Algorithm 4** author-title clustering algorithm

---

- 1: *query of first author name and two words of the title gathering a pool of documents repeated three times*
  - 2: *a  $n$ -gram string matching is used to compare two fields of a document pair*
  - 3: *A rule of transitivity increases a cluster :*
  - 4: **if** *R1 matches R2 and R2 matches R3 then*
  - 5:     *R1 matches R3*
  - 6: **end if**
- 

A variant of "Algorithm 4", proceeds in two steps, to build a key with bigrams (BGF) (see [33]). It has been called anchor method by [23]. The first step is the elaboration of a dictionary of bigrams from databases, theoretically, according to the alphabet (see its definition, above) we could find  $36 \times 36$ , hence 1,296 bigrams. After extracting their number of occurrences and excluding ambiguous bigrams such as *at, it, is* ...; we keep the 50 most frequent bigrams constituting the anchor dictionary *th, an, re, en, si, er, al, ce, es, pa, di, ...*. The second step transforms a title according to the present bigrams, from the dictionary, to build a key. For instance, according our example,

1. *Title = hybrid inhibitors of phosphatidylinositol 3-pinase (pi3k) and the mammalian target of rapamycin (mtor): design, synthesis, and superior antitumor activity of novel wortmannin-rapamycin conjugates*
2.  $K_{BGF} = id\ hi\ bi\ ph\ ph\ id\ na\ an\ he\ al\ nt\ es\ nt\ he\ si\ an\ er\ nt\ ac\ vi\ no\ ve\ an\ ni\ pa\ es$

The main concurrent approaches to fingerprinting are more related to definition of a space of descriptors and usage of similarity indices to find the closest semantically related documents. It is more or less based on famous document clustering techniques. In these approaches, as in fingerprinting approaches, heuristics are used to select features and to represent efficiently the content of each document.

Two widespread algorithms are devoted to this approach.

There is the Salton vector space approach (SVS), based on using distance metrics such as Euclidean distance between two documents (or the distance can be something else such as cosine measure or many other measures having the properties of a distance in a high-dimensional vector space cf. [8]). In our case, each dimension is associated to a word extracted from the titles and abstracts. The advantage of such a representation is that each dimension can receive a weight according to its presence in a document or the database, hence the famous TF-IDF. This means the product of the number of occurrence of a word in the database by the inverse number of occurrences in a current document. It weights positively the less frequent and locally-distributed words. The algorithm, "Algorithm 5" has three different parts. The first one builds a dictionary of words in the whole database and keeps the most frequent ones (more than 2 occurrences). The second part splits each document in the same way to make a lexical vector, keeping those already present in the dictionary. It builds a vector in the space of dictionary attributes. The third part chooses each document to classify by comparing it pairwise to each document of the target database. The similarity between such vectors of attributes can be computed using a cosine measure. If the result is greater than a convenient threshold (usually 0.95) then the current document is tagged as a duplicate.

The second approach is nearly the same as the previous one but the attributes are not words but collocations, and especially all 2-collocations and 3-collocations (see [6], [32]). We call it the collocation-similarity-based approach (CSB). As collocations are not so frequent it is more convenient for this approach to use a similarity coefficient to compare the set of collocations of a current document to another one. The algorithm can hence use a famous coefficient such as the Jaccard measure, computing ratio of intersection of attributes between two sets to the union of attributes. "Algorithm 6" details the routine.

---

**Algorithm 5** Salton Vector Space algorithm

---

**Require:** Databases files  $\{Dictionary\ building\}$ ;

- 1: **for** each document  $D_i$  in Databases **do**
- 2:     *select fields title and abstract, concatenate in a string  $S$*
- 3:     *Split  $S$  in single words {example: 'mammalian target of rapamycin' leads to the set 'mammalian','target','of','rapamycin'}*
- 4:     *sort list of words and store each word in a hash table  $HT$  with its rank  $r$ .*
- 5: **end for**

**Require:** A given Document  $\{Document\ Descriptor\ Building\}$ ;

- 6: *select fields title and abstract, concatenate in a string  $S$*
- 7: *Split  $S$  in all contiguous collocations of size 2 and 3, store in  $LW$*
- 8: *Set a vector  $V$  with  $\#V \leftarrow \#HT$*
- 9: **for** each word  $W$  in  $LW$  **do**
- 10:     **if**  $C$  belongs to  $HT$ , with rank  $r$  **then**
- 11:         *set  $V[r] \leftarrow 1$*
- 12:     **else**
- 13:         *set  $V[r] \leftarrow 0$*
- 14:     **end if**
- 15: **end for**

**Require:** Descriptor vectors  $V(i)$   $\{Document\ Comparison\ with\ a\ given\ document\ D_d\}$ ;

- 16:  $m \leftarrow \#HT$
- 17: *Threshold  $\leftarrow 0.95$*
- 18: **for** each vector  $V(c)$  of a document  $D(c)$  from the database **do**
- 19:     *compute a cosine similarity measure*

$$I = \frac{\sum_{i=1}^m (V_{ci} \cdot V_{di})}{\sqrt{\sum_{i=1}^m V_{ci}^2} \cdot \sqrt{\sum_{i=1}^m V_{di}^2}} \quad (1)$$

- 20:     **if**  $I > Threshold$  **then**
  - 21:         *set  $D(c)$  as duplicate of  $D(d)$*
  - 22:     **end if**
  - 23: **end for**
-

---

**Algorithm 6** Collocation Similarity-Based algorithm

---

**Require:** Databases files  $\{Dictionary\ building\}$ ;

- 1: **for** each documents  $D_i$  in Databases **do**
- 2:     *select fields title and abstract, concatenate in a string  $S$*
- 3:     *Split  $S$  in all contiguous collocations of size 2 and 3 {example:  
      'mammalian target of rapamycin' leads to set {'mammalian tar-  
      get','mammalian target of','target of', 'target of rapamycin'}}}*
- 4:     *sort list of collocations and store each collocation in a hash table  $HT$  with its rank  $r$ .*
- 5: **end for**

**Require:** A given Document  $\{same\ routine\ as\ previous\ as\ in\ SVS\ but\ split\ with\ collocations - Document\ Descriptor\ Building\}$ ;

**Require:** Descriptor vectors  $V(i)$   $\{Document\ Comparison\ with\ a\ given\ document\ D_d\}$ ;

- 6:  $m \leftarrow \#HT$
- 7:  $Threshold \leftarrow 0.95$
- 8: **for** each vector  $V_c$  of a document  $D_c$  from the database **do**
- 9:     *compute a Jaccard similarity measure*

$$J_{DC} = J(V_D, V_C) = \frac{|\{V_{D0}, \dots, V_{Dm}\} \cap \{V_{C0}, \dots, V_{Cm}\}|}{|\{V_{D0}, \dots, V_{Dm}\} \cup \{V_{C0}, \dots, V_{Ck}\}|} \quad (2)$$

- 10:     **if**  $I > Threshold$  **then**
  - 11:         *set  $D(c)$  as duplicate of  $D(d)$*
  - 12:     **end if**
  - 13: **end for**
- 

Other specific approaches try to investigate the properties of documents with original techniques, such as learning, but the results are not convincing. [3] proposes to learn the parameters (weights) between matrices of a generative model for string distance. For that they used forward-backward and EM algorithms. For each field they compute a distance measure hence making a vector of different measures they trained a SVM binary classifier to sort duplicates/non-duplicates. The result gives 100% of recall, precision is around 45%.

## 4 Our Language Models for Fingerprinting

### 4.1 Key definition

We make two keys from titles and authors which seems to be the most specific to scientific literature. In this way it may be applied to other documents (work documents, emails, blogs, ...) having generally a title and an author. Websites of course have a different document structure but this is not our purpose.

We decided to use two different methodologies for building a key. One is relatively intuitive, taking into account word-sequence and the author name (socio-semantic approach, or SSF). The second does not take this into account and is only based on compaction using the alphabet (monogram approach MGF), and a variant by sorting the key (sorted monogram approach, SMGF).

SSF proceeds as follows. First, title and first author name are concatenated into a single string S. Second, S is reduced to lower case. Finally, we keep only the first bigram of each word in the same order of occurrence of the words in the limit of the N first bigrams of the title. Practically, we take N=8. MGF is quite different in that way, the key is obtained only with the help of the alphabet. We check from left to right the first, and only the first, occurrence, keeping the order, of a letter from the alphabet. The key is reduced to lower case and no delimiters are taken into account. In the SMGF variant, the key is sorted alphabetically. In our example we thus obtain

- *AU - Ayral-Kaloustian, S Gu, JX Lucas, J Cinque, M Gaydos, C Zask, A Chaudhary, I Wang, JY Di, L Young, M Ruppen, M Mansour, TS Gibbons, JJ Yu, K*
- *TI - Hybrid Inhibitors of Phosphatidylinositol 3-Kinase (PI3K) and the Mammalian Target of Rapamycin (mTOR): Design, Synthesis, and Superior Antitumor Activity of Novel Wortmannin-Rapamycin Conjugates*
- $K_{SSF} = hy\ in\ of\ ph\ 3-\ (p\ an\ th\ ma\ ta\ ay$
- $K_{MGF} = hybridntorspal3keumcvwjg$
- $K_{SMGF} = abcdeghijklmnoprstuvw3$

These techniques achieve unique key extraction with a very compact representation using only 20 characters on average.

## 4.2 Algorithm

The algorithm has been written in Perl, and is quite fast. If the size of the test database is  $N$  and the size of the target database is  $M$  we need to compute a key for each  $(N+M)$  documents. This step has complexity  $O(N+M)$  in time. The second step relies upon hash lookup. Lookup operations in a 'map' data structure are guaranteed to have a time complexity that is at most logarithmic in the number of key-value pairs in the map. Hence for each test document we need a lookup of  $N \cdot \log(M)$ . Globally, the time complexity is  $O((N+M) + N \cdot \log(M))$ .

Complexity in space is not high. It requires the storage of  $(N+M)$  documents,  $(N+M)$  pointers to keys and hashkeys,  $(N+M)$  lists of words for titles (and author names). So the global complexity is  $O(N+M)$ . On a laptop with a pentium M processor and 2.3 GhZ clock,  $N=7,709$ ,  $M=12,658$ , the running time is 7 seconds. Space consumed is 254 Mb, around 5 times the size of the databases (the size on disk of the databases is 43 Mb ; 24 Mb for about  $N$  documents, 19 Mb for about  $M$  documents).

The algorithm is developed in two steps. A first step computes a key for each document (see previous chapter for details); a document can belong to the test or target database. The second step picks a document from the test database and its associated key, to compare it to all other keys for the target database. "Algorithm 7" describes the SSF approach. "Algorithm 8" describes the MGF approach. "Algorithm 9" describes the SMGF approach which varies a little bit from the MGF approach only by sorting the key.

---

**Algorithm 7** Socio-Semantic Fingerprinting algorithm

---

**Require:** Databases files {1st step, keys building};

```
1:  $N \leftarrow 8$ 
2: for each document  $D$  from the databases do
3:   select title ( $T$ ) and first author name ( $A$ )
4:   transform  $T$  and  $A$  in lower case
5:   split  $T$  into words, store in  $LW$ 
6:   for each words  $W_i$  from the  $LW$ , and  $i \leq N$  do
7:     select the first bigram of  $W$ , store in  $K$ 
8:   end for
9:   select the first bigram of  $A$ , store in  $K$ 
10: end for
11:  $K$  is stored in a hash table  $HT$ 
```

**Require:** given hash key of test document  $D(c)$ , hash table  $HT$  of target databases {2nd step, comparison};

```
12: if  $HT(K_{D(c)})$  return  $d$  then
13:   set  $D(c)$  as duplicate of  $D(d)$ 
14: end if
```

---

---

**Algorithm 8** Monogram Fingerprinting algorithm

---

**Require:** Databases files, Alphabet  $A$  {1st step, keys building}

```
1:  $Mirror_L A \leftarrow \#A$ 
2: for each document  $D$  from the databases do
3:   select title ( $T$ )
4:   transform  $T$  in lower case
5:   split  $T$  into letters of  $A$ , store in  $LA$ 
6:   for each letter  $L_i$  from  $LA$  do
7:     if  $Mirror_L A(L_i) \neq 0$  then
8:       store  $L_i$  in  $K$ 
9:        $Mirror_L A(L_i) \leftarrow 1$ 
10:    end if
11:  end for
12: end for
13:  $K$  is stored in a hash table  $HT$ 
```

**Require:**  $D(c)$ ,  $HT$  {2nd step, same as "algorithm 7" }

---



---

**Algorithm 9** Sorted Monogram Fingerprinting algorithm

---

**Require:** Databases files, Alphabet A {1st step, keys building}

- 1: *same as "algorithm 8"*
- 2: *sort K by alphabetical order*
- 3: *K is stored in a hash table HT*

**Require:** D(c), HT {2nd step, same as "algorithm 8"}

---

## 5 Evaluation

Two protocols were implemented, one using benchmark data (i.e., a gold standard), the extrinsic evaluation; the other using only datasets, the intrinsic evaluation.

### 5.1 Gold standard data

An offline step has been achieved to characterize a test set defined by hand. In this step we chose a set of documents in which we search for an exact duplicate. We fixed the size of this set to be 374 documents from the PM dataset. In this set, 202 documents were found to be exact duplicates. They settle a gold standard for future comparisons.

### 5.2 Intrinsic evaluation

The intrinsic evaluation is achieved within 3 protocols.

We call the first one, the Monte Carlo approach (MC). The test database is the gold standard set of 374 documents. No target database is defined and a tag is randomly chosen. The results are shown in Figure 2, almost 50% of precision and recall. It constitutes a kind of baseline for the rest of evaluation.

The second and third protocols use only PM database. In the second protocol, the PM database is split into two separate files with no redundancy; we call this protocol HPM (half PubMed evaluation). Theoretically the method should not find any duplicates. In practice it makes one mistake among the 3,855 documents. It finds that the following document, resembling a quasi-plagiat by the author, is a quasi-duplicate:

1. WOS sentence : *TI - Clonal relationship among Vibrio cholerae O1 El Tor strains isolated in Somalia. AB - One hundred and three Vibrio cholerae O1 strains, selected to represent the cholera outbreaks*

which occurred in Somalia in 1998-1999, were characterized by random amplified polymorphic DNA patterns, ribotyping, and antimicrobial susceptibility... AD - Dipartimento di Genetica e Microbiologia, Universita di Bari, Italy. FAU - Scrascia, Maria (PMID- 18774337)

2. PM sentence : *TI Clonal relationship among Vibrio cholerae O1 El Tor strains causing the largest cholera epidemic in Kenya in the late 1990s. AB Eighty Vibrio cholerae O1 strains selected to represent the 1998-to-1999 history of the largest cholera epidemic in Kenya were characterized by ribotyping, antimicrobial susceptibility, and random amplified polymorphic DNA patterns.... AU - Scrascia, Maria (PMID-16954285)*

The last intrinsic evaluation (third protocol) concerns usage of the PM database as test and target at the same time. We call this protocol FPM (Full PubMed evaluation). Theoretically it has to find all the documents as duplicates, and it does. So, the approach gets a rate of success quite reasonable at this step of evaluation.

	#Gold Duplicates	#Predicted duplicates	#True positives	#False positives	#False negatives	Precision ratio	Recall ratio
MC	202	203	94	109	108	0.463	0.465
HPM	0	1	0	1	3855	1.00	1.00
FPM	7709	7709	7709	0	0	1.00	1.00

Figure 2: Comparison between methods with a gold standard.

### 5.3 Comparison with other methodologies

In this benchmark, we tested 10 methodologies to compare documents. The initial goal is to compare pairwise documents between two different database, specifically PubMed and WebofScience to merge two corpora and delete duplicates.

Our techniques presented here are based on very compact language models to make a unique key for each document: socio-semantic fingerprint (SSF, "Algorithm 7"), monogram fingerprint (MGF, "Algorithm 8"), and the sorted monogram fingerprint (SMGF, "Algorithm 9"). Three approaches from the state of the art: the Salton vector space (SVS, "Algorithm 5"), collocation similarity-based (CSB, "Algorithm 6") and the bigram fingerprint

(BGF, "Algorithm 4"). Moreover, three simple key fingerprint approaches also were tested: author fingerprint (AF), title fingerprint (TF), author-review-date fingerprint (ARDF), and the modified title fingerprint (MTF) ("Algorithm 1").

We have three summarising scores. The basic methodologies AF, TF, ARDF and MTF are not efficient with recall score because we prefer recall maximization. SVS, CSB and BGF give good recall but precision is not so good as SSF and MGF. SMGF is catastrophic, pointing out the importance of the sequence order of the letters in a string. SVS has good results but unfortunately many more false positives than SSF. CSB differs from SSF only by 1 false positive in the CSB side. Nevertheless we can assert that SSF is at the top of the results, with MGF having no false positives and 2 false negatives less than SSF.

	#Gold Duplicates	#Predicted duplicates	#True positives	#False positives	#False negatives	Precision ratio	Recall ratio
<b>SSF</b>	202	192	192	0	10	1.00	0.95
<b>TF</b>	202	131	131	0	71	1.00	0.649
<b>MTF</b>	202	151	151	0	51	1.00	0.748
<b>AF</b>	202	314	202	112	0	0.643	1.000
<b>ARDF</b>	202	139	134	5	68	0.964	0.663
<b>MGF</b>	202	194	194	0	8	1.00	0.960
<b>SMGF</b>	202	295	197	98	5	0.668	0.975
<b>BGF</b>	202	196	193	3	9	0.985	0.955
<b>SVS</b>	202	212	201	11	1	0.948	0.995
<b>CSB</b>	202	193	192	1	10	0.995	0.950

Figure 3: Comparison between methods with a gold standard.

## 6 Conclusion

Duplication has been presented in this paper as a modern topic having applications such as merging databases or plagiarism detection. We aimed at merging two bibliographic record databases. In this area, cleaning is an important process, to eliminate redundant information. We presented several interesting and efficient algorithms from the state of the art. These algorithms can be improved.

Our language models, divided into two fingerprint key extraction, socio-semantic, and monogram keys, are able to get good scores in evaluation protocols compared to other algorithms. We get more than 95% recall with

100% precision.

## 7 Acknowledgments

Special thanks to Prof. Dr. Christian Meyer, at INRA Paris, a specialist in molecular biology and plant biology for cooperation about the TOR molecule.

## References

- [1] SF Altschul, TL Madden, AA Schaffer, RA Schffer, JH Zhang, Z Zhang, W Miller, and DJ Lipman. Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic Acids Res*, 25:3389–3402, 1997.
- [2] Y Bernstein and J Zobel. A scalable system for identifying co-derivative documents. In *Proceedings of the Symposium on String Processing and Information Retrieval*, pages 55–67. Springer, 2004.
- [3] M Bilenko and RJ Mooney. Adaptive duplicate detection using learnable string similarity measures. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-2003)*, pages 39–48, 2003.
- [4] D Bitton and DJ DeWitt. Duplicate record elimination in large data files. *ACM Transactions on Database Systems*, 8(2):255–265, 1983.
- [5] RS Boyer and JS Moore. A fast string searching algorithms. *Communications of the Association for Computing Machinery*, 20(10):762–772, 1977.
- [6] S Brin, J Davis, and H Garcia-Molina. Copy detection mechanisms for digital documents. In *In SIGMOD '95: Proceedings of the 1995 ACM SIGMOD international conference on Management of data*, pages 398–409, 1995.
- [7] AZ Broder. On the resemblance and containment of documents. In *Compression and Complexity of Sequences (SEQUENCES'97)*, pages 21–29. IEEE Computer Society, 1997.

- [8] A Chowdhury, O Frieder, D Grossman, and MC McCabe. Collection statistics for fast duplicate. *ACM Transactions on Information Systems*, 20, 2002.
- [9] F Deng. Approximately detecting duplicates for streaming data using stable bloom filters. In *In SIGMOD '06: Proceedings of the 2006 ACM SIGMOD international conference on Management of data*, pages 25–36. ACM Press, 2006.
- [10] AK Elmagarmid, PG Ipeirotis, and S Verykios. Duplicate record detection: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 19(1):1–16, 2007.
- [11] M Errami, JM Hicks, W Fisher, D Trusty, JD Wren, TC Long, and HR Garner. Dj vu - a study of duplicate citations in medline. *Bioinformatics*, 24(2):243–249, 2008.
- [12] A Feekin and ZX Chen. Duplicate detection using k-way sorting method. In *In the Proceedings of the 2000 ACM symposium on Applied Computing*, pages 323–327. ACM Press, 2000.
- [13] IP Fellegi and AB Sunter. A theory for record linkage. *Journal of the American Statistical Association*, 64(328):1183–1210, 1969.
- [14] CC Gong, YL Huang, XQ Cheng, and S Bai. Detecting near-duplicates in large-scale short text databases. In Ting Washio, Suzuki and Inokuchi, editors, *Advances in Knowledge Discovery and Data Mining, 12th Pacific-Asia Conference, PAKDD 2008, Osaka, Japan*, pages 877–883. Lecture Notes in Computer Science Springer, 2008.
- [15] P Goyal. Duplicate record identification in bibliographic databases. *Information Systems*, 12(3):239–242, 1987.
- [16] MA Hernandez and SJ Stolfo. The merge/purge problem for large databases. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 1995.
- [17] TB Hickey and DJ Rypka. Automatic detection of duplicate monographic records. *Journal of Library Automation*, 2(12):125–142, 1979.
- [18] TC Hoad and J Zobel. Methods for identifying versioned and plagiarised documents. *Journal of the American Society for Information Science and Technology*, 54:203–215, 2003.

- [19] JA Hylton, JH Saltzer, and FR Morgenthaler. Identifying and merging related bibliographic records. MIT LCS Masters Thesis, 1996.
- [20] DE Knuth and VR Pratt. Fast pattern matching in strings. *SIAM Journal on Computing*, 6(2):323–350, 1977.
- [21] LI Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.
- [22] M Madhavaram, DL Ali, and M Zhou. Integrating heterogeneous distributed database systems. *Computers and Industrial Engineering*, 31(1-2):315–318, 1996.
- [23] U Manber. Finding similar files in a large file system. In *Proceedings of the USENIX Winter 1994 Technical Conference*, pages 1–10, 1994.
- [24] A Monge and Ch Elkan. An efficient domain-independent algorithm for detecting approximately duplicate database records. In *Proceedings of the ACM SIGMOD international conference on Management of data*, 1997.
- [25] K Muthmann, A Lser, WM Barczynski, and F Brauer. Near-duplicate detection for web-forums. In *Proceedings of the International Database Engineering & Applications Symposium*, 2009.
- [26] HB Newcombe, JM Kennedy, SJ Axford, and AP James. Automatic linkage of vital records. *Science New series*, 130(3381):954–959, 1959.
- [27] Pl Paskalev and A Antonov. Intelligent application for duplication detection. In *Proceedings of International Conference on Computer Systems and Technologies - CompSysTech'2006*, 2006.
- [28] M Potthast and B Stein. New issues in near-duplicate detection. In Lars Schmidt-Thieme Christine Preisach, Hans Burkhardt and Reinhold Decker, editors, *Studies in Classification, Data Analysis, and Knowledge Organization; Data Analysis, Machine Learning and Applications*, pages 601–609. Springer, 2008.
- [29] M Ridley. An expert system for quality control and duplicate detection in bibliographic databases. *Program*, 26(1):1–18, 1992.
- [30] N Shivakumar and H Garcia-molina. The scam approach to copy detection in digital libraries. In *Proceedings of the Second Annual Conference on the Theory and Practice of Digital Libraries*, 1995.

- [31] TF Smith and MS Waterman. Identification of common molecular subsequences. *Journal of molecular biology*, 147:195–197, 1981.
- [32] JJ Tamilselvi and V Saravanan. Detection and elimination of duplicate data using token-based method for a data warehouse: A clustering based approach. *International Journal of Dynamics of Fluids*, 5(2):145–164, 2009.
- [33] ZP Tian, HJ Lu, WY Ji, AY Zhou, and Z Tian. An n-gram-based approach for detecting approximately duplicate database records. *International Journal on Digital Libraries*, 3(4):325–331, 2002.
- [34] RR Yager and A Rybalov. On the fusion of documents from multiple collection information retrieval systems. *Journal of the American Society for Information Science*, 49:1177–1184, 1998.
- [35] M.I Yampolskii and A.E Gorbonosov. Detection of duplicate secondary documents. *Nauchno-Tekhnicheskaya Informatsiya*, 1(8):3–6, 1973.